2012 IEEE 12th International Conference on Data Mining

Self-Training with Selection-by-Rejection

Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080
{yan.zhou2, muratk, bhavani.thuraisingham}@utdallas.edu

***Abstract*—Practical machine learning and data mining problems often face shortage of labeled training data. Self-training algorithms are among the earliest attempts of using unlabeled data to enhance learning. Traditional self-training algorithms label unlabeled data on which classifiers trained on limited training data have the highest confidence. In this paper, a self-training algorithm that decreases the disagreement region of hypotheses is presented. The algorithm supplements the training set with self-labeled instances. Only instances that greatly reduce the disagreement region of hypotheses are labeled and added to the training set. Empirical results demonstrate that the proposed self-training algorithm can effectively improve classification performance.**

***Keywords*-semi-supervised learning, self-training**

I. Introduction

Semi-supervised learning (SSL) techniques are developed for learning tasks where the amount of labeled data is insufficient for learning a hypothesis with good generality. The common assumption is that labeled data and unlabeled data are independent and identically distributed, and some unlabeled data can be mapped to a specific label with a high likelihood in a predefined framework. Frameworks differ in how unlabeled data is blended in with labeled data. Most frameworks fall into the following categories: confidence-based, model-based, graph-based, and multi-view based [1]. A detailed survey of semi-supervised learning techniques can be found in Zhu [2]. In this paper, we focus on one of the simplest confidence-based SSL techniques, self-training.

Self-training [3], [4], also known as self-teaching, is one of the earliest techniques using both labeled and unlabeled data to improve learning. Given a set of labeled data L and unlabeled data U, self-training proceeds as follows: train a classifier h using L, and classify U with h; select a subset $U' \subset U$ for which h has the highest confidence scores; add U' to L and remove U' from U. Repeat the process until the algorithm converges. In this paper, we consider semi-supervised learning as a search problem. The goal is to find subsets of unlabeled data and their proper labels to compensate for the lack of labeled data. Inclusion of the selected unlabeled data would effectively correct the current decision boundary. Unlabeled data improves learning if: 1) it supplies additional information on the true decision boundary; 2) it retains the overall data distribution; and 3) it introduces bounded noise. Without the information of the true labels, it is difficult to foretell if a selected pool of unlabeled instances hold the aforementioned properties. We find it helpful to examine the opposite aspect of using unlabeled data—cases where the above properties are violated and therefore labeling unlabeled data is not rewarding.

Case 1 Selected unlabeled data does not add more information on the decision boundary. There are two possible consequences when the selection is used for training:

1.) there is very little impact on the classification of given labeled data, yet the classification accuracy deteriorates in general;
2.) it misguides the training process and causes costly maladjustment of the decision boundary, which in turn backfires and causes a significant increase in misclassification of the labeled data.

Case 2 A biased selection of unlabeled data favors one particular class of data. This happens when the classifier used to label the unlabeled data is biased. In this case, overfitting becomes more likely.

Case 3 An overwhelming amount of noise is introduced to the training set during the process of expansion. If unbounded noise is introduced, eventually classification of labeled data will become no better than random guessing. As a result, adding unlabeled data would actually hurt the classifier's performance.

Therefore if the inclusion of an unlabeled data pool has a marginal impact on classification of labeled data, it is not clear whether we are approaching the true decision boundary. On the other hand, if the addition of unlabeled data leads to much greater misclassification of the labeled data, we can safely conclude that the unlabeled data with the current assigned labels are harmful, either because they are incorrectly labeled or they have been sampled disproportionately. Thus we need to assign different labels to the examples or rebalance the data distribution in the selection. This is the foundation of our self-training algorithm.

With a classifier h, we let h label a subset of the unlabeled data, but invert the labels assigned by h. In other words, if h predicts "+" for an unlabeled instance X, we assign "–" to X as its label in a binary classification problem. In multi-classification problems, if h predicts ℓ for an unlabeled instance X, we assign $\overline{\ell}$ to X where $\overline{\ell}$ is any legitimate

1550-4786 2012

DOI 10.1109/ICDM.2012.56

label but ℓ. We then train a new classifier on the training set that includes the newly labeled unlabeled data. If the new classifier holds its classification accuracy on the labeled data, we select a different subset from the unlabeled pool and restart the process. If, on the other hand, the new classifier fails largely on the labeled data, we restore the labels of the selected unlabeled instances assigned by h and add the selection to the training set. We repeat the process until we can no longer identify such a subset of unlabeled data to add to the training set. We refer to this technique as *selection by rejection*. This proof-by-contradiction style of data selection sets our algorithm apart from other semi-supervised learning algorithms.

In the rest of the paper, we first present the algorithm of our self-training technique in Section II. In Section III, we present the theoretical foundation of this technique. We demonstrate the empirical results in Section IV and discuss the related work in Section V. Finally, we conclude our work in Section VI.

II. Self-Training Algorithm

We now discuss the design of our self-training algorithm. Given a set of labeled data L and a set of unlabeled data U, a classifier h is trained on L initially. h is used to label all the unlabeled data U. The algorithm then randomly selects and assesses N subsets $U_1, \ldots, U_N$ of U each of which, if included for training, is likely to improve classification. One of the subsets will be selected and added to the training set. h is retrained on the expanded training set. The process repeats until stopping criteria are met. Detailed algorithm is given in Figure 1. There are several issues we have to resolve: First, how to obtain a set of unlabeled data for which the current classifier is likely to provide correct labels? Second, how to find the most effective subset among the N subsets? And finally, when does the algorithm halt?

A. Candidate Unlabeled Subsets

In theory if we sample enough data, the sample noise introduced as a result of misclassification by a given hypothesis would not shadow the benefits of an augmented labeled data set [5]. In practice it is nearly impossible to estimate how much is enough for two reasons: 1) we do not know exactly the size of the hypothesis space $|\mathcal{H}|$; 2) there are often not enough samples (including the unlabeled ones) for us to draw enough samples to compensate for the noise. Therefore we need better "engineering" for data selection to ensure low sampling noise. The brute force approach would explore each possible combination of unlabeled data in the search space. Like many other brute force approaches, its computational intractability calls for a better search strategy that can greatly reduce the search effort. A good starting point would be sets of unlabeled data each of which is highly likely labeled correctly by the classifier. For each unlabeled data point, we estimate its distance to the current decision boundary. An unlabeled data point that is at least δ distance away from the decision boundary would be considered as a good candidate. The larger the value of δ, the more likely the assigned label is correct. It has been shown that error variance is greatest near decision boundary [6]. Therefore, δ must be large enough to avoid noise caused by great uncertainty in the vicinity of the decision boundary. On the other hand, an overly conservative δ would bias toward outliers. In a binary classification problem, a data point x is considered to be on the decision boundary if $\frac{p_+}{p_-} = 1$ where p_+ is the probability of x being positive, and p_- is the probability of x being negative. To facilitate multi-class classification problems, we estimate the distance to a decision boundary as the negation of the entropy of class probabilities (*line* 9 in *Algorithm* 1).

In each round of self-training we randomly sample N subsets of unlabeled data from U that are δ distance away from the current decision boundary (*lines* 11–14 in *Algorithm* 1). As self-training progress to the next round, the unlabeled data set U is updated using random sampling with replacement according to the weight of each unlabeled data point. Unlabeled instances that have been selected for training in previous rounds will have weights that have decayed exponentially (*lines* 23–24 in *Algorithm* 1). The higher the decay rate, the less likely the same instance would be added to the training set more than once. Next, we discuss how to choose the most influential subset from the N subsets of preselected unlabeled data.

B. Selection by Transformation

There are two reasons that our algorithm begins with multiple candidate subsets in each round. First, not all unlabeled data, despite correct labeling, is helpful in improving supervised learning. This phenomenon has been emphasized and extensively researched in the field of active learning. Second, unlabeled data points that are "far away" from the decision boundary could be outliers. Therefore, simply adding unlabeled data that is situated in the classifier's high confidence zone is not always a good idea. To find a set of unlabeled data that is truly informative, we resort to our aforementioned *selection by rejection* strategy.

In essence, we transform the problem of selecting a set of unlabeled data to the problem of rejecting the same set of unlabeled data with its class membership inverted. Let $\hat{h}$ denote the classifier trained on $L \cup U$, where U has been labeled using the classifier trained on the current labeled data set (*line* 15 in *Algorithm* 1). As discussed in the previous section, N subsets of unlabeled data are randomly selected with older samples decay in weight exponentially in each round. For each of the preselected N subsets of unlabeled data U_j ($j = 1, \ldots, N$), we create a subset U'_j by inverting the label ℓ_j to ℓ'_j for each data point in U_j, where ℓ_j is the label predicted by the classifier trained on the labeled data set, $\ell'_j \in \mathcal{L} \backslash \{\ell_j\}$ and $\mathcal{L}$ denotes the class space. A classifier

h_j is trained on each $L \cup U'_j \cup (U - U_j)$. $\hat{h}$ and $h_{j|\forall j \in [1,N]}$ are used to estimate Δd_j—the maximum disagreement between $\hat{h}$ and $h_{j|\forall j \in [1,N]}$ on L_0—the original labeled data plus unlabeled data points added with high confidence (*line* 27 in *Algorithm* 1). The theoretical implication of Δd_j and its computation (given in Equation 1) will be discussed in the next section. The subset $U_r \in \{U_1, \ldots, U_N\}$ that maximizes Δd_j is "rejected". *Lines* 16–22 in *Algorithm* 1 describe this process. After reverting the labels of data in U_r, all instances in U_r will be added to the training set for training a new classifier (*line* 26 in *Algorithm* 1). By adding more informative data to the training set in each round, we gradually guide the decision boundary to approach the true decision boundary.

Theoretically, optimal subsets can be identified only when N approaches positive infinity. In practice, N needs to be large enough to include subsets whose membership inversion would cause significant increase in misclassification of the originally labeled data. It is not surprising that our algorithm does not require a very large N since the pre-selection step ensures that most pre-selected unlabeled data, if mislabeled, will cause drastic classification performance drop. This is because the selected data points are located in areas where class membership is less uncertain.

C. Stopping Criteria

The last issue we need to resolve is when the algorithm should stop. We let the algorithm continue running until there is no unlabeled data that carries significant boundary information left. In each round, we resample the unlabeled data set U. As more and more unlabeled data points that satisfy the distance requirement have been labeled at least once, the average weight $\overline{w}$ of data points in U will decline at an increasingly slower pace. In the extreme case where all unlabeled data points are equally important (informative) and are labeled all at once, the average rate will drop to α—the rate of weight decay—after the first round of self-labeling and the algorithm can halt at that point. In our algorithm, we check if $\overline{w} < c \cdot \alpha$ (*line* 29 in *Algorithm* 1), where c, a positive constant, is the knob that controls how soon the algorithm terminates. A larger c terminates the algorithm sooner than a smaller c.

To summarize we present the following nutshell version of our self-training algorithm. First, we assign labels to the unlabeled data with the current classifier; next we randomly select, as many as processing time permits, N subsets of unlabeled data that are δ distance away from the decision boundary; we select from the N subsets the most effective subset to add to the training set using selection-by-rejection. The process repeats until no more qualified unlabeled data has not been added to the training set.

There are several constants presented in the proposed algorithm. c is the knob constant that determines how soon the algorithm halts, α is the decay rate for the weights of unlabeled data, N is the number of pre-selected subsets at the beginning of each round, and finally δ, the distance threshold, is set to the median of distances of all unlabeled data to the current decision boundary. Selection of the values of the constants is discussed in a later section.

Input: L, L_0—a set of labeled data
U—a set of unlabeled data
T—a set of test data
c—knob constant, α—decay rate
Output: Predictive accuracy on T

1: **for all** i such that $x_i \in U$ **do**
2: $\mathbf{w}[x_i] = 1.0$
3: $\overline{w}$ is the average weight over all $x_i \in U$
4: **repeat**
5: Train classifier h on L
6: $U_\delta = \emptyset$ // unlabeled data δ distance away from decision boundary
7: **for all** i such that $x_i \in U$ **do**
8: h predicts label $\arg\max_{\ell_k \in \ell} p(\ell_k|x_i)$ for x_i
9: $\mathbf{d}[x_i] = \sum_{\forall \ell_k \in \ell} p(\ell_k|x_i) \cdot \log p(\ell_k|x_i)$
10: $\delta = \widetilde{\mathbf{d}}$ // Median of $\mathbf{d}$—distances to decision boundary
11: **for all** i such that $x_i \in U$ **do**
12: **if** $\mathbf{d}[x_i] > \delta$ **then**
13: $U_\delta = U_\delta \cup \{x_i\}$
14: Randomly sample (with weight decay) N subsets $U_1, \ldots, U_N$ from U_δ
15: Train classifier $\hat{h}$ on $L \cup U$
16: **for** $j = 1$ **to** N **do**
17: $U'_j = \emptyset$
18: **for all** i such that $x_k \in U_j$ **do**
19: $x'_k =$SETLABEL$(x_k, \ell'_k \in \ell \backslash \{\ell_k\})$
20: $U'_j = U'_j \cup \{x'_k\}$
21: Train classifier h_j on $L \cup U'_j \cup (U - U_j)$
22: $U_r = \arg\max_{U_j} \Delta d_j(\hat{h}, h_j, L_0)$
23: **for all** i such that $x_i \in U_r$ **do**
24: $\mathbf{w}[x_i] = \mathbf{w}[x_i] \cdot \alpha$
25: Update $\overline{w}$
26: $L = L \cup U_r$
27: $L_0 = L_0 \cup \{x \in U_r | h\ highly\ confident\ for\ x\}$
28: Resample U with replacement based on $\mathbf{w}$
29: **until** $\overline{w} < c \cdot \alpha$
30: $h^* = \arg\min_h error(h, L_0)$
31: **return** $1 - error(h^*, T)$

Figure 1. Self-Training

III. Theoretical Analysis

This section presents the theoretical justification of the self-training algorithm. We first define a sampling process where there is a noisy oracle that increases the mislabeling rate p by introducing additional random noise to the

classification noise η. The action of the oracle models the label inversion process in our self-training algorithm. We show that regardless of η, our mislabeling rate p can be confined to less than $\frac{1}{2}$, thus we can learn an ϵ-good hypothesis (*Theorem 3.5 and 3.8*) [5]. Naturally it follows that the probability of learning an ϵ-good hypothesis increases as we turn off the noise oracle (*Theorem 3.6*). In other words, we show that using the sampling process in our self-training algorithm, we can learn an ϵ-good hypothesis with a monotonically increasing probability for any arbitrary $\epsilon \in (0, \frac{1}{2})$. We also argue that by using label inversion and minimizing the disagreement region, we can select unlabeled examples that are not only more likely labeled correctly but more informative compared to the rest in the unlabeled data set. Next, we formally define the sampling process.

Given a set of labeled data L and a set of unlabeled data U, a hypothesis h is formulated on L. Let $\eta = error_D(h)$ denote the error of the hypothesis h under distribution D. We define the following sampling process for our self-training algorithm:

Randomly draw M samples from D and let them pass through an oracle h before being added to the training set. With probability $\eta = error_D(h)$ the oracle returns an incorrectly labeled example $(x, \overline{\ell})$, and with probability $1-\eta$ it returns a correctly labeled example (x, ℓ). In addition, each example x has to pass through a noise oracle that randomly inverts the decision of h from ℓ to $\overline{\ell}$ or from $\overline{\ell}$ to ℓ with a probability of η_f.

A hypothesis h is ϵ-good if it has an error bounded by ϵ with a bounded confidence δ, that is,

$$Pr[error(h) > \epsilon] < \delta.$$

In other words, the probability that h has an error greater than ϵ is less than δ. A hypothesis is ϵ-bad if its error rate is greater than ϵ. We now prove that with the above sampling process we can learn an ϵ-good hypothesis with a monotonically increasing probability for any arbitrary $\epsilon \in (0, \frac{1}{2})$ in each round of our self-training algorithm.

Let p be the probability that an example x is mislabeled. Therefore, p is the probability that h correctly labels x but the noise oracle inverts the label to the incorrect one, plus the probability that h mislabels x and the noise oracle does not invert the label. Thus,

$$p = \eta(1-\eta_f) + (1-\eta)\eta_f.$$

We first consider the case where $0 < \eta < \frac{1}{2}$.

Lemma 3.1: If $0 < \eta < \frac{1}{2}$, choose an $\eta_f \in (0, \frac{1}{2})$. It follows that $0 < p < \frac{1}{2}$.

Proof:

$$\begin{aligned} p &= \eta(1-\eta_f) + (1-\eta)\eta_f \\ &= \eta(1-2\eta_f) + \eta_f \\ &< \frac{1}{2}(1-2\eta_f) + \eta_f \\ &= \frac{1}{2} \end{aligned}$$

■

Lemma 3.2: If $0 < \eta < \frac{1}{2}$, then $\eta < p < \frac{1}{2}$.

Proof: According to Lemma 3.1, $p < \frac{1}{2}$. We only need to prove that $\eta < p$.

$$\begin{aligned} p - \eta &= \eta(1-\eta_f) + (1-\eta)\eta_f - \eta \\ &= \eta_f - 2\eta\eta_f \\ &= \eta_f(1-2\eta) \\ &> 0 \end{aligned}$$

■

Let m be the number of samples drawn from D, and $|\mathcal{H}|$ be the size of a finite set of hypotheses.

Theorem 3.3: [5] Given a mislabeling rate $p < \frac{1}{2}$ (See Lemma 3.1), if we draw a sample of size

$$m \geq \frac{2}{\eta^2(1-2p)^2} \ln(\frac{2|\mathcal{H}|}{\delta})$$

from D, the probability that an η-bad hypothesis minimizes disagreement with the sample is at most δ.

We simple substitute p for the upper bound of the noise rate and η for ϵ in the noisy PAC model.

Lemma 3.4: Let $\eta' < \eta$ and $p' = \eta'(1-\eta_f) + (1-\eta')\eta_f$. If a sample S of size m is sufficient for learning an η'-good hypothesis with the mislabeling rate p, S is sufficient for learning an η'-good hypothesis with the mislabeling rate p'.

Proof:

$$\begin{aligned} p &= \eta(1-\eta_f) + (1-\eta)\eta_f \\ &= \eta(1-2\eta_f) + \eta_f \end{aligned}$$

Given that $\eta_f < \frac{1}{2}$, therefore $1 - 2\eta_f > 0$. Thus p decreases monotonically with η. Therefore, $p' < p$. According to Theorem 3.3, it requires a smaller sample to achieve the same classification error with a lower mislabeling rate. Therefore, if m is large enough to learn an η'-good hypothesis with the mislabeling rate p, m is sufficient to learn an η'-good hypothesis when the mislabeling rate is reduced to p'. ■

Theorem 3.5: For any $\epsilon \in (0, \frac{1}{2}) < \eta$ and $\delta \in (0, 1)$, let $n = \lceil \log_2(\frac{1}{\epsilon}) \rceil$ and $k = (\frac{\epsilon}{\eta})^{\frac{1}{n}}$, our self-training algorithm outputs a hypothesis with $err_D(h) \leq \epsilon$ with probability of at least $1 - \delta$ after n rounds. The number of samples drawn in each round i is

$$M = \frac{2}{(k^i\eta)^2(1-2p)^2} \ln(\frac{2|\mathcal{H}|}{\delta}).$$

Proof: According to Theorem 3.3 and Lemma 3.4, in each self-training round i, we draw M samples to reduce $err_D(h_i)$ by a factor of $\frac{1}{k}$. After n rounds of self-training, we reduce $err_D(h)$ to $k^n\eta = \frac{\epsilon}{\eta}\eta = \epsilon$. The sample complexity is

$$\mathrm{O}(\ln(\frac{1}{\epsilon})\frac{1}{\epsilon^2(1-2p)^2} \ln(\frac{2|\mathcal{H}|}{\delta})).$$

■

M sets the bound of the number of examples we need to draw in each round in order to reduce the learning error by a fixed factor of k. So far we have proved that with a sample mislabeling rate of p our self-training algorithm outputs an ϵ-good hypothesis after n rounds of drawing M samples. We now introduce our *selection by rejection* sampling technique to the self-training procedure. Let L_i be the set of labeled data in the i^{th} round of our self-training procedure. Let $\mathcal{H} = \{h_1{}^i, h_2{}^i, \ldots, h_N{}^i\}$ be a finite set of N hypotheses. Each $h_j{}^i \in \mathcal{H}$ is learned from $L_i \cup U_j$ of M samples randomly drawn from D (M is given in Theorem 3.5). Examples in U_j are labeled by h_i but with η_f probability the labels are inverted by the noise oracle. From Theorem 3.5 we know that for each $h_j{}^i \in \mathcal{H}$

$$d(h_j{}^i, h^*) = Pr_D[h_j{}^i(x) \neq h^*(x)] \leq k^i\eta$$

where h^* is the target concept with $err_D(h^*) = 0$. In other words, all the hypotheses in $\mathcal{H}$ are at least as good as h_i. Now, suppose we turn off the noise oracle and repeat the above process. Let $\hat{\mathcal{H}}$ be a set of N hypotheses in which each hypothesis is learned from $L_i \cup U_j$—the same set of samples drawn from U but with $\eta_f = 0$. According to Lemma 3.2 and Theorem 3.5,

$$d(\hat{h}_j^i, h^*) = Pr_D[\hat{h}_j^i(x) \neq h^*(x)] \leq k^i\eta.$$

Let

$$\Delta d_j = \max_{p \in \{1,\ldots,N\}} d(h_p{}^i, h^*) - d(\hat{h}_j^i, h^*).$$

The quantity Δd_j consists of two parts:

$$\Delta d_j = \Delta d(\hat{h}_j^i, h_j{}^i|h^*) + \Delta d(h_u{}^i, h_v{}^i|h^*) \qquad (1)$$

where $\Delta d(\hat{h}_j^i, h_j{}^i|h^*)$ denotes the disagreement between hypotheses $\hat{h}_j^i$ and $h_j{}^i$ with respect to h^* and $\Delta d(h_u{}^i, h_v{}^i|h^*)$ denotes the disagreement between $h_u{}^i \in \mathcal{H}$ and $h_v{}^i \in \mathcal{H}$ with respect to h^* where $u, v \in \{1, \ldots, N\}$ and $u \neq v$. In other words, $\Delta d(\hat{h}_j^i, h_j{}^i|h^*)$ assesses whether the labels inverted by the noise oracle are causing the decision boundary to drift away from the true decision boundary; and $\Delta d(h_u{}^i, h_v{}^i|h^*)$ signifies whether one set of samples drawn from U is more informative than the other. The former has a natural affinity with semi-supervised learning where the goal is to draw samples from the unlabeled data set to enhance the labeled data set without introducing too much noise; while the latter demonstrates the idea of active learning in which the goal is to draw the most informative samples to minimize the disagreement region between h^* and hypotheses in the version space. Let

$$\overline{U} = \arg\max_{U_j \in \{U_1,\ldots,U_N\}} \Delta d_j$$

be the set of unlabeled samples drawn from D that maximizes Δd_j. Thus $\overline{U}$ (with inverted labels) is our target to "reject". The unlabeled examples in $\overline{U}$ have two properties: 1.) it is more likely that they are labeled correctly by the oracle—because inverting the labels causes the decision boundary to move farther away from the true decision boundary (that is, larger $\Delta d(\hat{h}_j^i, h_j{}^i|h^*)$); 2.) they are located in the maximum disagreement region between two hypotheses in $\mathcal{H}$. Therefore, examples in $\overline{U}$ are not only more likely labeled correctly by the oracle but more informative compared to examples in $U_j \neq \overline{U}$.

In practice, we do not know M—the bound of the samples to be drawn in the i^{th} round. Instead, we just use all available unlabeled data to train $\hat{h}_j^i$, that is, $\hat{h}$ in *Algorithm* 1. Also notice that in Equation 1, h^* is unknown. In theory h^* needs to be known to compute this difference, which is impossible in general. We resolve the difficulty by estimating the difference on L_0—the original labeled data plus data labeled with high confidence. Although this estimate is rough, it is sufficient to manifest the deterioration in classification performance when the labels are incorrectly inverted.

Theorem 3.6: In each self-training round i, $\overline{U} \in \{U_1, \ldots, U_N\}$ is selected as a result of *selection by rejection*. Let $L_{i+1} = L_i \cup \overline{U}$ where $\overline{U}$ is labeled by h_i with $\eta_f = 0$. The probability that we learn an ϵ-bad hypothesis h_{i+1} in the next round is monotonically decreasing, that is,

$$Pr[d(h_{i+1}, h^*) > \epsilon] < Pr[d(h_i, h^*) > \epsilon].$$

Sketch of Proof According to Theorem 3.5, $\overline{U} \in \{U_1, \ldots, U_N\}$ is sampled to monotonically reduce $err_D(h_i)$ to a fraction k of $err_D(h_i)$ in the presence of sample noise that consists of classification noise introduced by h_i and random noise introduced by the noise oracle. When forming the training set for round $i+1$, the noise oracle is turned off (because inverted labels are restored) and the sample noise is the sole contribution from h_i. Therefore the proof follows naturally from Lemma 3.4 and Theorem 3.5. □

So far we have only considered the case where $0 < \eta < \frac{1}{2}$. When $\eta > \frac{1}{2}$, we simply increase the noise rate in the noise model to $\frac{1}{2} < \eta_f \leq 1$. In addition, all the unlabeled examples in $\overline{U}$ (after being labeled by h_i) need to have their labels inverted before they are added to the training set. Similar lemmas and theorems are developed as follows.

Lemma 3.7: if $\frac{1}{2} < \eta \leq 1$, $p \in (0, \frac{1}{2})$ if $\frac{1}{2} < \eta_f \leq 1$.

According to Lemma 3.7, when $\eta = error_D(h) > \frac{1}{2}$, we need to increase the noise rate of the noise oracle so that the combined error p is bounded below $\frac{1}{2}$. Let $\overline{\eta} = 1 - \eta$.

Theorem 3.8: Given $1 \geq \eta > \frac{1}{2}$ and a mislabeling rate $p < \frac{1}{2}$ (See Lemma 3.7), if we draw a sample of size

$$m \geq \frac{2}{(\overline{\eta})^2(1-2p)^2} \ln(\frac{2|\mathcal{H}|}{\delta})$$

from D, the probability that an $\overline{\eta}$-bad hypothesis minimizes disagreement with the sample is at most δ.

Theorem 3.9: In each self-training round i, let $\overline{U}_f$ be the set of unlabeled examples selected (through rejection). Let

$L_{i+1} = L_i \cup \overline{U}_f$ where $\overline{U}_f$ is labeled by h_i but having each label inverted. The probability that h_{i+1} is an ϵ-bad hypothesis decreases monotonically in the next round, that is, $Pr[d(h_{i+1}, h^*) > \epsilon] < Pr[d(h_i, h^*) > \epsilon]$.

Theorem 3.9 states that even when $error_D(h_i) > \frac{1}{2}$, self-training is still effective if we let h_i label the selected unlabeled samples but having all the labels inverted afterwards. For multi-class problems, we can rank the class labels according to the prior distribution, and have a label inverted to the one with the highest rank other than itself.

IV. Experimental Results

We tested our self-training algorithm on 17 UCI [7] data sets, 13 of which are binary classification problems, and 4 of which are multi-class classification problems. We chose as our baseline classifier the logistic regression algorithm in the latest distribution of Weka [8]. We also tested the standard self-training algorithm with logistic regression as the baseline and the state-of-the-art semi-supervised support vector machine algorithm on the same data sets for comparison. In the binary classification cases, our self-training algorithm improved over all data sets, and it clearly outperformed the standard self-training algorithm [3] and the SVMLight [9] implementation with LOQO solver [10], [11] in most cases. In addition, the SVM method seemed to be less consistent compared to the other two algorithms. In the cases of multi-class classification, our self-training algorithm again improved over all data sets. It outperformed the standard self-training algorithm in all four cases and the univerSVM [12] implementation of the semi-supervised SVM method in two out of four cases. Parameters in all SVM implementations were carefully selected for each data set as suggested so that the algorithm would converge. Parameter values in various ranges were tried. Little variations were observed whenever the algorithm converges.

A. Binary Classification

We selected 13 binary classification problems from the UCI repository. Table I shows the number of labeled, unlabeled, and test instances of each data set. For small data sets, we reserve at most 10% of the data as labeled data, the rest is used as test data. For very large data sets, including *sick*, *mushroom*, and *kr-vs-kp*, a very small percentage (5%, 0.5%, 1%, respectively) of data was set for training. Unlabeled data was selected as a subset of each test set. Unlabeled data and test data are the same for small data sets. For large data sets, 20% of the test data is used as unlabeled data.

Table II shows the classification results when the initial labeled data is stratified. It presents in order the predictive accuracy of the logistic regression algorithm (LR) when only the labeled data is used for training, the standard self-training algorithm, our proposed self-training algorithm, and the semi-supervised SVM method. Our self-training results improved over all data sets. In addition, it output better results than both of the other two algorithms in 10 out of 13 cases. The semi-supervised SVM improved on 8 of the 13 data sets, but failed to improve over the rest of the 5 data sets. The proposed self-training algorithm demonstrated better results than the SVM method on 10 of the 13 data sets. The standard self-training algorithms improved on 6 of the 13 data sets. It was outperformed by our algorithm in 11 out of the 13 cases. The advantage of our self-training algorithm becomes more obvious when the initial labeled data is not stratified, as shown in Table III. Our self-training algorithm again improved over all data sets, while the SVM method improved on 5 data sets and the standard self-training improved on 7 data sets. Our self-training algorithm performed better than SVM and the standard self-training algorithm on 12 of the 13 data sets. All the results reported in this paper are the average over 100 runs. Since the semi-supervised SVM always labels data, for fair comparison we excluded cases where no performance change was observed simply because no data or very little data was labeled. Constants are set as follows: $N = 10$, $c = 4$, and $\alpha = 0.125$. We also tried other constant values. Our observations are as follows: large N values slow down the algorithm without much reward. Since each unlabeled subset is carefully selected to avoid the high uncertainty region near the decision boundary, the advantage of large Ns is trivialized. Large c values lead to more rounds of labeling and thus more noise that eventually will hurt the classification accuracy. Large α values reduce the chance of sample replacement, making successive labeling rounds more dependent on the preceding ones. When the data set is small, we prefer a small α value.

For the four data sets that have a very low sample-to-dimension ratio, we tested again with a higher labeled-to-unlabeled data ratio (> 4). The results are shown in Table IV and Table V. Our self-training algorithm improved significantly over all four data sets and demonstrated clear advantage over the other two algorithms. The standard self-training algorithm had marginal improvement on all four data sets. The SVM method showed less improvement on all

Table I
Statistics of the selected UCI data sets.

Data set	Labeled	Unlabeled	Test
Vote	44	391	391
Sonar	21	187	187
Sick	189	716	3583
Mushroom	51	1614	8073
Labor	6	51	51
Kr-vs-kp	32	632	3164
Ionosphere	36	315	315
Hepatitis	16	139	139
Heart-stat	27	243	243
Diabetes	77	691	691
Credit-g	100	900	900
Credit-a	69	621	621
Breast-w	70	629	629

Table II

CLASSIFICATION ACCURACIES ON 13 UCI DATA SETS WHEN THE INITIAL LABELED DATA IS STRATIFIED. THE FIRST COLUMN IS FOR LOGISTIC REGRESSION WHEN ONLY LABELED DATA IS USED; THE SECOND COLUMN IS THE STANDARD SELF-TRAINING ALGORITHM (STD-SELF-TRAINING), THE THIRD COLUMN IS FOR OUR SELF-TRAINING ALGORITHM, AND THE LAST ONE IS FOR THE SEMI-SUPERVISED SVM METHOD (SS-SVM).

Data set	Labeled Only	Std-Self-Training	Self-Training	SS-SVM
Vote	0.932±0.022	**0.951 ±0.098**	**0.953±0.015**	**0.948±0.019**
Sonar	0.657±0.056	0.638 ±0.097	**0.696±0.051**	**0.704±0.052**
Sick	0.956±0.007	0.954 ±0.094	**0.961±0.006**	0.894±0.016
Mushroom	0.937±0.024	0.930 ±0.097	**0.962±0.016**	**0.947±0.022**
Labor	0.683±0.115	**0.785 ±0.102**	**0.777±0.110**	**0.804±0.100**
Kr-vs-kp	0.818±0.069	0.795 ±0.124	**0.858±0.063**	0.665±0.132
Ionosphere	0.806±0.034	**0.809 ±0.090**	**0.833±0.027**	**0.816±0.032**
Hepatitis	0.769±0.051	**0.772 ±0.102**	**0.814±0.030**	**0.810±0.054**
Heart-statlog	0.757±0.044	0.744 ±0.089	**0.786±0.035**	**0.767±0.018**
Diabetes	0.741±0.022	0.739 ±0.077	**0.752±0.019**	0.736±0.023
Credit-g	0.696±0.022	**0.700 ±0.073**	**0.714±0.018**	0.679±0.023
Credit-a	0.829±0.023	**0.849 ±0.088**	**0.849±0.014**	0.826±0.016
Breast-w	0.942±0.017	0.936 ±0.097	**0.952±0.012**	**0.966±0.005**

Table III

CLASSIFICATION ACCURACIES ON 13 UCI DATA SETS WHEN THE INITIAL LABELED DATA IS NOT STRATIFIED. THE FIRST COLUMN IS FOR LOGISTIC REGRESSION WHEN ONLY LABELED DATA IS USED; THE SECOND COLUMN IS FOR THE STANDARD SELF-TRAINING ALGORITHM, THE THIRD COLUMN IS FOR SELF-TRAINING, AND THE LAST COLUMN IS FOR THE SEMI-SUPERVISED SVM METHOD (SS-SVM).

Data set	Labeled Only	Std-Self-Training	Self-Training	SS-SVM
Vote	0.931±0.027	**0.951±0.097**	**0.954±0.011**	0.913±0.040
Sonar	0.640±0.069	0.618 ±0.098	**0.683±0.064**	**0.679±0.061**
Sick	0.955±0.008	0.953±0.096	**0.959±0.007**	0.907±0.042
Mushroom	0.928±0.031	0.924±0.100	**0.958±0.019**	0.923±0.032
Labor	0.592±0.180	**0.770 ±0.119**	**0.708±0.152**	**0.719±0.123**
Kr-vs-kp	0.799±0.073	**0.809±0.120**	**0.841±0.065**	0.537±0.051
Ionosphere	0.804±0.039	**0.808±0.093**	**0.825±0.033**	**0.805±0.046**
Hepatitis	0.763±0.051	**0.773±0.102**	**0.807±0.033**	0.650±0.083
Heart-statlog	0.751±0.043	0.740±0.087	**0.777±0.043**	0.717±0.067
Diabetes	0.742±0.018	0.734±0.080	**0.755±0.015**	0.655±0.054
Credit-g	0.694±0.019	**0.703±0.073**	**0.714±0.014**	0.675±0.026
Credit-a	0.829±0.022	**0.848±0.087**	**0.850±0.014**	**0.830±0.027**
Breast-w	0.939±0.020	0.937±0.097	**0.952±0.012**	**0.950±0.021**

data sets, and when the initial labeled data was not stratified, the SVM method had either no improvement or marginal improvement on the four data sets.

B. Multi-class Classification

We also tested on four multi-classification data sets from the UCI repository. The four data sets are *zoo*, *waveform-5000*, *vowel*, and *vehicle*. For each data set, we allocated 10% for training, and the rest for testing. Unlabeled data is again selected as a subset of each test set. Table VI shows the classification results of the four settings—labeled only, the standard self-training algorithm, our self-training algorithm, and the SVM method—when the labeled data set is stratified. Our self-training algorithm successfully improved over all data sets. It also outperformed the SVM method on two data sets. The SVM method improved the classification performance on two data sets, but largely hurt the performance on the other two data sets. The standard self-training algorithm did not improve on any of the data sets. When the initial labeled data was not stratified, we observed similar results as shown in Table VII.

V. RELATED WORK

Hearst [13] and Yarowsky [3] use self-training for resolving word sense disambiguation. Riloff et al. [14] use self-training through multi-level bootstrapping for selecting extraction patterns. They also use self-training to identify subjective nouns [4]. Rosenberg et al. [15] build a self-training model for object detection from images. The basic assumption these techniques have developed upon is that an unlabeled instance can be labeled for training if the underlying classifier has high confidence in its prediction. Our self-training algorithm gradually improves a classifier through a similar boot-strapping style of self-teaching. However, we do not consider all the unlabeled data that the classifier has confidently labeled equally informative. Some of the unlabeled data could be outliers, some others could be correctly labeled but carries very little additional information on the true decision boundary. Our self-training algorithm employs a random selection procedure that rules out these two cases as we discussed earlier.

A group of probabilistic approaches have also been proposed for using unlabeled data in supervised learning. Unlabeled data, due to its large amount, can be used to identify

Table IV
CLASSIFICATION ACCURACIES ON 4 SMALL UCI DATA SETS WHEN THE INITIAL LABELED DATA IS STRATIFIED.

Data set	Labeled Only	Std-Self-Training	Self-Training	SS-SVM
Sonar	0.756±0.084	0.747 ±0.120	**0.814±0.078**	**0.782±0.082**
Labor	0.831±0.069	**0.892 ±0.125**	**0.935±0.062**	**0.908±0.075**
Ionosphere	0.869±0.044	**0.871 ±0.104**	**0.903±0.040**	**0.896±0.046**
Hepatitis	0.817±0.077	**0.828 ±0.114**	**0.885±0.071**	**0.843±0.093**

Table V
CLASSIFICATION ACCURACIES ON 4 UCI DATA SETS WITH LOW SAMPLE-TO-DIMENSION RATIOS WHEN THE INITIAL LABELED DATA IS NOT STRATIFIED.

Data set	Labeled Only	Std-Self-Training	Self-Training	SS-SVM
Sonar	0.742±0.073	**0.746 ±0.119**	**0.802±0.067**	0.722±0.104
Labor	0.836±0.077	**0.880 ±0.123**	**0.931±0.065**	0.829±0.111
Ionosphere	0.868±0.051	**0.871 ±0.103**	**0.899±0.047**	**0.883±0.051**
Hepatitis	0.812±0.079	**0.826 ±0.115**	**0.882±0.073**	**0.827±0.086**

Table VI
CLASSIFICATION ACCURACIES ON 4 MULTI-CLASS UCI DATA SETS WHEN THE INITIAL LABELED DATA IS STRATIFIED.

Data set	Labeled Only	Std-Self-Training	Self-Training	SS-SVM
Zoo	0.677±0.072	0.638±0.141	**0.740±0.059**	**0.843±0.047**
Waveform-5000	0.851±0.006	0.850±0.086	**0.855±0.005**	0.794±0.013
Vowel	0.507±0.043	0.449±0.060	**0.526±0.036**	0.263±0.037
Vehicle	0.676±0.029	0.656±0.077	**0.695±0.025**	**0.720±0.022**

Table VII
CLASSIFICATION ACCURACIES ON 4 MULTI-CLASS UCI DATA SETS WHEN THE INITIAL LABELED DATA IS NOT STRATIFIED.

Data set	Labeled Only	Std-Self-Training	Self-Training	TSVM
Zoo	0.633±0.127	0.595±0.172	**0.707±0.103**	**0.787±0.083**
Waveform-5000	0.850±0.006	0.849±0.085	**0.855±0.005**	0.793±0.012
Vowel	0.492±0.040	0.444±0.059	**0.510±0.037**	0.262±0.036
Vehicle	0.660±0.036	0.647±0.075	**0.683±0.030**	**0.718±0.021**

the mixture components of a generative model. Nigam et al. [16] apply EM with mixture models for using unlabeled data. They demonstrate promising results in the field of text classification. However, when the mixture model assumption is incorrect, mixture model based semi-supervised learning can in fact hurt classification performance [17]. Several techniques have been proposed to address this issue [18], [19].

Graph-based semi-supervised learning methods treat labeled and unlabeled data as graph nodes. The edge between a pair of nodes entails similarity between the two corresponding data points. Blum and Chawala [20] propose a graph mincut solution to semi-supervised learning. Discussions on other graph-based approaches, including direct Markov random fields, Gaussian random fields and harmonic functions, manifold regularization, and many others can be found in Zhu and Goldberg's work [1].

Blum and Mitchell present co-training [21] that takes advantage of data sets that have two independent and redundant views. The algorithm improves supervised learning by letting two classifiers trained on the two different views label data for each other. Some other co-training style semi-supervised learning techniques employ multiple learners on single-view data sets [22], [23].

In this paper, we focus on self-training and therefore only compare our algorithm to the self-training SSL algorithms.

VI. CONCLUSIONS

We present a robust self-training algorithm that improves learning by selecting and labeling more informative unlabeled data. A random selection procedure, namely *selection by rejection*, is presented to guide the search for informative unlabeled data subsets. The experiment results demonstrate that our proposed algorithm is in general more reliable and more effective than the standard self-training algorithm and the semi-supervised SVM method. Its classification performance is also more consistent across all datasets than the other two methods.

ACKNOWLEDGMENT

This work was partially supported by The Air Force Office of Scientific Research MURI-Grant FA-9550-08-1-0265 and Grant FA9550-12-1-0082, National Institutes of Health Grant 1R01LM009989, National Science Foundation (NSF) Grant Career-CNS-0845803, NSF Grants CNS-0964350, CNS-1016343, CNS-1111529, and CNS-1228198, Army Research Office Grant 58345-CS.

REFERENCES

[1] X. Zhu and A. Goldberg, *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, 2009.

[2] X. Zhu, "Semi-supervised learning literature survey," 2006. [Online]. Available: http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

[3] D. Yarowsky, "Unsupervised word sense disambiguation rivalling supervised methods," in *Proceedings of the Thirty-third Annual Meeting of the Association for Computational Linguistics*, 1995, pp. 189–196.

[4] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 25–32.

[5] D. Angluin and P. Laird, "Learning from noisy examples," *Machine Learning*, vol. 2, pp. 343–370, 1988.

[6] J.-M. Park and Y.-H. Hu, "On-line learning for active pattern recognition," *IEEE Signal Processing Letters*, vol. 3, pp. 301–303, 1996.

[7] C. J. Merz and P. M. Murphy, "UCI repository of machine learning databases," 1998.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[9] K. Scheinberg, "An efficient implementation of an active set method for svms," *Journal of Machine Learning Research*, vol. 7, pp. 2237–2257, 2006.

[10] R. Vanderbei, "Loqo: An interior point code for quadratic programming," Princeton University, Tech. Rep. SOR-9415, 1998.

[11] A. Smola, "pr_loqo optimizer," http://www.kernel-machines.org/code/prloqo.tar.gz.

[12] F. Sinz, R. Collobert, J. Weston, and B. L., "Universvm: Support vector machine with large scale cccp functionality," http://www.kyb.tuebingen.mpg.de/bs/people/fabee/universvm.html.

[13] M. A. Hearst, "Noun homograph disambiguation using local context in large text corpora," in *University of Waterloo*, 1991, pp. 1–22.

[14] E. Riloff and R. Jones, "Learning dictionaries for information extraction by multi-level bootstrapping," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999, pp. 474–479.

[15] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *The Seventh IEEE Workshop on Applications of Computer Vision*, 2005.

[16] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine Learning*, vol. 39, pp. 103–134, 2000.

[17] F. G. Cozman, I. Cohen, M. C. Cirelo, and E. Politcnica, "Semi-supervised learning of mixture models," in *ICML-03, 20th International Conference on Machine Learning*, 2003, pp. 99–106.

[18] B. Shahshahani and D. Landgrebe, "The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon," *IEEE Trans. On Geoscience and Remote Sensing*, vol. 32, pp. 1087–1095, 1994.

[19] D. Miller and H. Uyar, "A mixture of experts classifier with learning based on both labelled and unlabelled data," in *NIPS 9*. MIT Press, 1997, pp. 571–577.

[20] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 19–26.

[21] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998, pp. 92–100.

[22] S. Goldman and Y. Zhou, "Enhancing supervised learning with unlabeled data," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 327–334.

[23] Z.-H. Zhou and M. Li, "Semi-supervised regression with co-training," in *Proceedings of the 19th international joint conference on Artificial intelligence*, 2005, pp. 908–913.